Design Proposal for PICos DSP ALU

Leiqing Cai
Electrical and Computer Engineering
University of Virginia
lc9ac@virginia.edu

Qin Qing
Electrical and Computer Engineering
University of Virginia
qq3za@virginia.edu

Ashley Morse
Electrical and Computer Engineering
University of Virginia
alm3tw@virginia.edu

1 INTRODUCTION

The Portable Instruments Company (PICo) is calling for a bid for an embedded Digital Signal Processor design in the FreePDK 45nm technology. The arithmetic and logical operation desired by the DSP system includes, but not limited to, AND, OR, PASS, ADD, SUB, SHIFT, NOP on two 16-bit inputs. In this paper, we propose a high-performance design that fulfills the required functionality while exhibiting high computational throughput, low energy consumption, and low area cost.

2 DESIGN DESCRIPTION

2.1 Overview

In the DSP system, a 3-bit control input s selects the operation performed on the two 16-bit inputs A and B. A correspondence between the control and the function selected is shown in Table 1.

As shown in Table 1, the system also support an 8×8 multiplication operation in addition to the basic features. When multiplication is selected, the ALU treats the lowest 8-bit of the inputs as two unsigned integers and compute the product and output it as a 16-bit signal.

All inputs are latched into registers, as well as the output of the Arithmetic Logic Unit (ALU). This allows the DSP system to run in a pipelined fashion. By doing so, the system supports as much as 2.78 billion basic operations per second.

Table 1: ALU functions for different opcodes

Opcode	Operation	Function
000	NOP	$Out_n = Out_{n-1}$
001	AND	$Out = A\&B$
010	OR	$Out = A\|\|B$
011	PASS A	$Out = A$
100	ADD	$Out = A + B$
101	SUB	$Out = A - B$
110	SHIFT	$Out = A << B$
111	MULT	$Out = A \times B$

2.2 Adder/Subtractor

To built a 1-bit Full Adder (FA), We decided to use a 24-transistor mirror adder implementation. We then directly connected 16 FA to form the carry-ripple adder, without inverting back the carry-out of each single FA. By doing so, we had to invert inputs on odd bits and outputs on even bits, but it saves 16 inverter delay.

By observing the fact that only one of the addition and subtraction operation is needed for each clock cycle, the adder block can be shared by both functions. According to the rules of two's complements,

$$A - B = A + \bar{B} + 1$$

We should feed the adder with A and $\bar{B}$ as the data inputs and "1" as the carry-in to obtain $A - B$. This explain why we purposely choose $s_0 = 0$ for addition and $s_0 = 1$ for subtraction, as by doing so, we could feed B XOR s_0 to the second operand of the adder, s_0 and to the carry-in.

2.3 AND/OR/PASS

AND and OR are implemented using pass-transistor logic. PASS is implemented with transmission gates.

2.4 Shifter

We implemented the SHIFT function using a barrel shifter with 2 stages. In the first stage, each bit is shifted by either

2 or 0 position to the left. In the second stage, each bit is shifted by either 2 or 1 position to the left. In each stage, input bits are connected to the two possible destination bits by pass gates, but only one of them will be on at any moment. The purpose of using pass gate is to minimize energy consumption, as no current will be drawn from the power supply for pass gates.

To recover voltage swing, we added a buffer for each output bit before it is passed on to the next logic element.

2.5 ALU Mux

During a clock period, all combination elements described above do its work, but only 1 out of the 8 signals will be selected as the output for the particular cycle. Although we call it "ALU Mux", we finally decided to implemented it as decoder-controlled transmission gates ("Tri-state Buffer"). For each one of the 16 bits, there are 8 different sources of inputs, each of them connects to the output with a transmission gate. Only 1 of the 8 transmission gates will be on at any moment. Therefore, a 3-to-8 decoder is needed to translate the select signal s to the control of the transmission gates. Figure 2 gives a hint of what the ALU Mux looks like. In the figure, 4 of the 8 bits are connected. The remaining bits should be connected similarly.

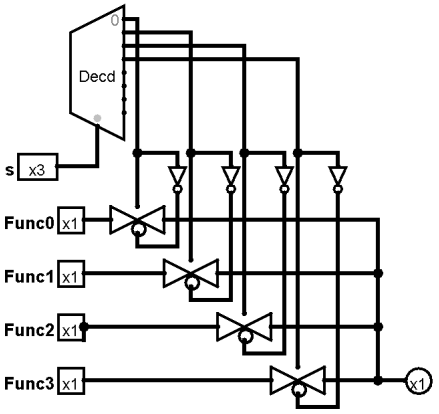

Figure 1: Decoder-Tx-based implementation of ALU Mux

3 INNOVATION

Among all the metrics, we first sought ways to minimized the worst case delay. Once it was determined, the goal then turned to pushing down energy consumption and area cost.

3.1 Critical Path Optimization

The critical path of the ALU consists of the Adder-Subtractor and the 16-bit 8:1 Mux. Section 2.2 and 2.5 describe the structure of our Adder-Subtractor and ALU Mux.

For the ADD/SUB block, the longest delay happens when carry ripples from the least significant bit (LSB) to the most significant bit (MSB). In such cases, the total delay of the Adder-Subtractor is (assuming $t_s > t_c$)

$$t_{addsub} = t_{xor} + 15t_c + t_s$$

in which t_{xor} is the propagation delay of the XOR gate, and t_c and t_s are the propagation delays of the carry and sum stage. Here, t_c dominates the delay formula. We therefore chose mirror adder over Static CMOS Adder and Manchester Carry-Chain Adder as it has a relatively low t_c and also saves number of transistors. Another advantage that the mirror adder provides is shorter stack of transistors, which lowers the Elmore delay of each FA. In forming the ripple-carry adder, instead of inverting back each inverted carry-out, we inverted the inputs on the even bit locations so that those 16 inverter delays are cut while functionality remains intact.

In our original design of the ADD/SUB block, we used a 16-bit inverter and a 16-bit 2:1 Mux to generate the correct value for the second operand of the adder. By digging this a little bit further, we found that the logic here is

$$Input_2 = B\bar{s_0} + \bar{B}s_0 = B \oplus s_0$$

Therefore, we replaced the Inverter-Mux combination with a 16-bit XOR gate. By doing so, the critical path of addition and subtraction becomes the same, and we are allowed to focus only on optimizing the XOR gate to achieve faster operation.

For the XOR gate, we ended up using a 6-transistor-per-bit implementation, shown in Figure 2.

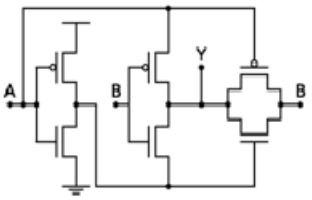

Figure 2: 6-transistor implementation of XOR[1]

We also tweaked the sizing of the adder. We chose the ratio of PMOS W/L to NMOS W/L to be 1:1, as a (90nm) characteristic PMOS has similar pulling force as a characteristic NMOS in this technology. Increasing PMOS width slightly decreases total delay but the additional area surpasses the gain in the speed. We also tried to use logic effort to size the 16-FA chain, so that the output capacitance over the 16 stages is four times the input capacitance. This approach saves 10 ps (out of about 335 ps) in worst case scenario, but the downside is also dramatic increase in area and energy consumption. Therefore, in our final design, all FA block has a size of 1.

The ALU Mux is yet another influential delay contributor. Our original design uses 7 2:1 Mux to create the 8:1 Mux. This incurs three times of the static CMOS Mux delay, and $20 \times 7 = 140$ units of area per bit.

We observed that the Mux does not start doing useful work until its input had all been stable. Because it has to wait for a long Adder-Subtractor delay, we can use transmission gates with on/off control pre-computed. Since these controls are determined only by s, they may be computed in

[1]Figure from RedCat Devices:
http://www.redcatdevices.it/projects/riflash

much shorter time than the ADD/SUB delay. The transmissions gates then just sit there and pass the signals as soon as they are ready from the ADD/SUB.

Our Decoder-Tx-based Mux is able to pass the signal in 19 ps, whereas the original Mux delay is 75 ps. Also, the area per bit is reduced to 86 units per bit.

By the reasoning above, we believe our critical path is reliable and has a vastly minimal delay metric.

3.2 Buffering

One issue with our original design topology is that the output of the registers $regA$, $regB$ and $regS$ need to drive a very large load, as they'll present as the inputs for each combinational element. RC delay in this case could be long. Therefore, we decided to insert buffer at these nodes. We used double inverters as the buffer. The first one has a size of 3 and the second one has a size of 9. We picked the value based on pre-calculated sizing (3 to 4) that minimizes the delay. We then use simulation to determine the optimal buffer sizes. We chose to use two inverters rather than two buffers in order to reduce an inverter delay.

3.3 Energy and Area Optimization

For the AND and the OR gate, starting out as static CMOS gates, there wasn't much room for optimizing. Still, we cut the number of transistors by using pass-transistor logic. The following netlist implements a 1-bit AND gate with only 3 transistor, while attaining full voltage swing. (A and B are 1-bit input and o is 1-bit output)

```
M0 (A B o VSS) NMOS size=size
M1 (A B_ o VDD) PMOS size=size
M2 (VSS B_ o VSS) NMOS size=size
```

$M0$ and $M1$ forms a transmission gate. It passes full swing AB when B is high. When B is low, $M2$ pulls down the output to ground since NMOS passes good "0". Due to the fact that we need an additional inverter to generate $\bar{B}$, it costs us a total of 5 characteristic-sized transistor for each bit of the AND gate, reducing one from the static CMOS implementation.

The same spirit applies to the OR gate.

In another thing worth noticing is the sizing of the registers. In our original design, we used size of 2 for $regA$, $regB$ and $regO$, and size of 8 for $regS$. This results in high values in energy and area. Since we added buffer in our revised design, we changed all the registers to minimum size in order to reduces these two metrics.

3.4 Trade Offs

One trade-off we through about was to use lower voltage on the components that are not on the critical path to decrease their energy consumption. One reason that we kept all of the components powered at the same voltage level was to avoid building level converters, which would increase the complexity of the circuit. Another reason is that after a simple analysis of our energy simulation, we figured out that the energy consumed by components not on the critical path is only about 8.4% of the total energy consumption. This result deepened our belief that we should not mess around with the supply voltage.

3.5 Extra Functionality

In the DSP system we present, we have an additional multiplication function. Some details of the functionality was explained in section 2.1. The 8×8 multiplier is constructed using four 4×4 Wallace-Tree multipliers. A high-level Wallace tree diagram for the 8×8 multiplier is shown in Figure 3.

Each multiplicand (A and B) is divided to upper 4 bits and lower 4 bits ($AHigh$, $ALow$, $BHigh$, $BLow$). Each dot in Figure 3 represents a bit. Each row is a 8-bit product generated by a 4×4 Wallace-Tree multiplier. The first and last column corresponds to $ALow \times BLow$ and $AHigh \times BHigh$, respectively, while the middle two rows corresponds to $ALow \times BHigh$ and $AHigh \times BLow$. Each tall box represents a FA while the shorter box represents a Half Adder (HA). After one pass, each bit location from 4 to 15 has at most two values to add, thus an adder of length 12 suffices.

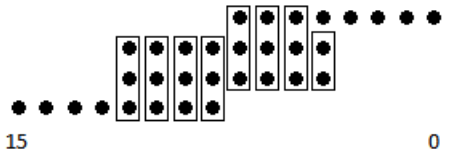

Figure 3: Wallace Tree for 4 8-bit product

4 RESULTS

4.1 Metrics

Table 2 shows the metrics we were able to achieve. "Delay" is the clock period we picked and minimized that still ensure the proper function of the DSP system even in the worst case scenario. "Energy consumption" is the measured average energy per-cycle consumed by the DSP system. "Area" is the total sum of device widths. The "Combined" metrics is the product of the above three quantities.

For energy simulation, A alternates between $0x$AAAA and $0x5555$ per cycle. $B = 0x$AA55. s counts up from $0b000$ to $0b111$. Clock period is 360 ps. The rationality of this clock period will be addressed in section 4.3.

When referring to "the DSP system", buffers that drives the primary data inputs, A and B, and the control inputs s are not included.

Table 2: Metrics for our DSP system

Metrics	Value
Delay	3.600×10^{-10} s
Energy	7.034×10^{-13} J
Area	4.457×10^{-4} m
Combined	1.129×10^{-25} s J m

The worst-case delay data shows that devices will be able to operate on a 2.78 GHz clock frequency.

4.2 Metrics Breakdown

Table 3 shows the breakdown of each metrics of our DSP system. The Area in this table is shown in the unit of "ch", which stands for the width of a characteristic transistor (90 nm).

Table 3: Breakdown of Metrics

Component	**Delay**	**Energy**	**Area**
Register		178.74	1020
ADD/SUB	1	190.23	1392
Mux	1	76.97	1376
Buffer	1	205.30	840
SHIFT	1	45.47	132
AND	1	0.30	80
OR	1	6.40	80
PASS	1	0.10	32
Unit	**ps**	**fJ**	**ch**

4.3 Comments on Energy Simulation

The energy consumption measured by simulation heavily depends on the inputs. Most of them are made arbitrary so that randomness is reflected when average is calculated. However, clock frequency is fixed throughout a simulation. When selected clock frequency is too low comparing to the maximum allowable frequency, leakage power becomes dominant while dynamic power becomes minuscule.

5 CONCLUSION

By presenting the data of maximum allowable frequency, average energy consumption and active area, we believe that our design beautifully fits the PICo's desire of the high performance and cost effectiveness. The multiplication feature enable hardware acceleration for some digital signal processing needs.

In addition, we are able to provide a very clean and easily manageable netlists of the design, which is helpful for future maintenance or upgrade.

In summary, our product is the right choice for PICo!